CSE 380
Lab #7
Spring 2017

Objective
In this lab, you will incorporate all of the concepts you learned in labs #1 through #6 to design a game.

Description
Write an ARM assembly language program that implements the game *Wee Dig Dug*, as described below.

Synopsis
Wee Dig Dug will be a modified version of the classic arcade game *Dig Dug*. The video game *Dig Dug* was developed and published in 1982 by Namco in Japan. It was licensed outside of Japan by Atari. The objective of *Wee Dig Dug* is the same as *Dig Dug* with some minor revisions. *Wee Dig Dug* will have the game start with level one and the player in the middle of the game board. There will be random areas that will not have "dirt" on the board. It is inside these areas that the enemies will reside. Points are earned by navigating through the dirt as well as by defeating enemies. Once the enemies have been cleared, a new level starts. Additional points are awarded for each new level. The game ends after two minutes of play or after all the player's lives have been used up. If the game ends after two minutes, points are awarded for each of the unused lives that are left. *Wee Dig Dug* cannot move through a border that surrounds the board (Z). If you are not familiar with the game, there is an online version of the game you may play to learn the mechanics of the game. You will find the link on the course website. Note that in the online version, enemies are defeated by inflating them with three pumps of air to destroy them. In *Wee Dig Dug*, a single pump of air destroys an enemy. In addition, there are no rocks in *Wee Dig Dug*. This is a simplification over the online version of *Dig Dug*.

The Board
A sample text-based board game board is shown below. The legend to the right of the board defines the ASCII characters on the board, but you ay choose whatever characters you prefer as long you clarify the meaning of the characters. The board should be 19 columns wide by 15 rows high (excluding the border).

```
ZZZZZZZZZZZZZZZZZZZZ
Z                  Z
Z                  Z
Z##################Z
Z##### x ###########Z
Z##################Z
Z##################Z
Z##################Z
Z######### > #######Z
Z##################Z
Z##################Z
Z##### B ###########Z
Z##################Z
Z##################Z
Z########### x #####Z
Z##################Z
ZZZZZZZZZZZZZZZZZZZZ
```

Legend

```
Z:  Unbreakable Wall
#:  "dirt"
>:  Player (facing right)
x:  Small Enemy
B:  Big Enemy
```

Basic Flow of Program
- Before the game begins, there should be a description of how the game is played. Instructions should be given on what the keystrokes are to control the game.

- The user should be given instructions on how to start the game. For example, "Press 'g' to begin."
- Once the game starts, the screen should be populated with the board. Excluding the border of the board (Z) there are 15 rows and 19 columns.
- The player will start in the center of the board and will have varying characters that indicate which direction the player is facing. For example, <, >, ^, and v may indicate the player is facing left, right, up, or down respectively.
- The player can navigate through the dirt.
- The player cannot move faster than the refresh rate of the game.
- There should be three (3) enemies, starting in random positions on the board. One will move faster than the other two.
- The two types of enemies should be different and should be indicated on the start screen.
- Enemies cannot move through dirt. They can only navigate through tunnels.
- The player can inflate an enemy that is in front of the player with an air pump. A single pump of air destroys an enemy. In other words having the hose from the air pump come in contact with the enemy is enough to destroy the enemy.
- When attempting to inflate an enemy, the air hose cannot be placed through dirt. To connect the air hose to the enemy, there must be a tunnel between the player and the enemy. In other words, the air host can NOT be placed through dirt.
- The level increases once all enemies are cleared from the board.
- The player's score should be indicated on the screen. Scoring is indicated in the scoring section shown below.
- The player loses a life by coming into contact with an enemy.
- When the game starts, the player has four (4) lives.
- The LEDs should be illuminated to reflect the number of lives left. When a life is lost, the life count on the LEDs must be decremented by turning one of the LEDs off. When all of the LEDs are off, the game is over.
- The seven-segment display should display the level the user is on. The program starts with level 0, and moves to level 1 once game play begins.
- Once the game play begins, the 7-segment display displays the level (1), four LEDs are illuminated to give the user four lives, the RGB LED should be green indicating the game play is active, and enemies start to move.
- The external interrupt can be used to pause the game. The game screen should state the game is paused. When the game is paused, the RGB LED should be blue.
- When game ends, the user should be prompted to either play again or quit. When the game ends, the RGB LED should be red.

Scoring

- The point system is based on the following:
  - Game starts with zero (0) points
  - For every space in the dirt (#) that the player navigates through, 10 points are awarded. If the player is navigating through a tunnel, no point are award for those space in the tunell.
  - Defeating a normal enemy is worth 50 points.
  - Defeating a hard enemy is worth 100 points.

Hardware Utilized

- **Serial Port** – *User Input & Display*
  - Used to accept the user's input and display the game.
- **LEDs** - *Number of Lives*
  - The LEDs will represent the number of lives remaining. You will start the game with four lives (all LEDs on). Each time a life is lost, an LED is turned off.

- **RGB LED** - *Game Status*
  - The RGB LED should be illuminated to display the status of the game. Before game play actually starts, it should be white. When the game is being played, it should be green. When the game is paused, the RGB LED should be blue. When a shot is fired, it should blink red. When the game ends, it should be purple.
- **Seven –Segment Display** - *Level*
  - The seven-segment display shows the current level the user is on. The initial level is 1. Before play actually starts, it should display level 0. It should increment by one with each completed level.
- **Momentary Push Button (P0.14)** – *Pause Button*
  - The momentary push button (P0.14) is the only push button which may be configured to trigger an interrupt. This will be used as a pause button. When pressed, your game must stop all movement, and not accept further keyboard input. When the button is pressed again, game play will resume. When the game is paused you must display "PAUSE" at the top of the board to tell the user the game has been paused. The "PAUSE" notification should disappear when the game is resumed.
- **Timers** – *Control Game Movement*
  - The timer is used to control the game speed and refresh the display. During the first level (level 1), characters should move two times per second.
  - With each level increase, the period should decrease by 0.1 seconds. Once the rate becomes 0.1 seconds, I tis capped. Additional levels may be entered, but the speed of the game will not increase.
  - The timers control the time limit for the entire game, which is two (2) minutes.

Grading
This project will count twice, carrying the weight of two (2) labs.

Prelab Writeup
You must submit a prelab writeup describing how you will solve the problem to the TAs BEFORE you log in to work on lab #7. This will be graded as part of lab #7. Failure to submit this prior to logging in to start lab #7 will result in a grade of zero (0) on this component of the lab. To stay on track and finish the lab in time, this report should be submitted no later than Monday, April 10 or Tuesday, April 11. When you get to the lab on April 10 or April 11 for your regularly scheduled lab session, you will NOT be allowed to enter the lab if you do not have this report with you or have not submitted it beforehand.

The Target
You will be running your program on the ARM processor.

Partners
You will work with one partner in this lab. Your partner *MUST* be the same partner you had in the previous lab.

Extra Credit
Extra credit will be awarded to those groups which submit the lab early. If you submit before 5:00 PM on Friday, April 28, 2017, you will receive ten (10) points extra credit. If you submit before 5:00 PM on Friday, April 21, 2017, you will receive twenty (20) points extra credit. In order to be eligible for the extra credit, your lab must work properly.

Documentation

Your program must be clearly commented, and documentation must also be provided. The documentation must follow the guidelines covered in lecture (found on the *Lectures* webpage of the course website). Your comments should describe what *each* section of your program does. The comments must describe what *each* section of your program does. Hardcopies of your code should **NOT** be submitted.

Demonstrations & Submissions

You will demonstrate your game to your TA in one of your regularly scheduled lab sessions during the week of May 8, 2017. Your source code must be submitted online using the submit command (submit_cse379 *filelist*) on *timberlake.cse.buffalo.edu* no later than 5:00 PM on Friday, May 5, 2017. Late submissions will NOT be accepted. Your documentation must be submitted in class on Monday, May 8, 2017. If your documentation is NOT submitted in class on May 8, 2017 or before, it will NOT be accepted.